Institute for Scientific Computing Research

ISCR Subcontracts Research Summaries

Subcontract Research Summaries

The ISCR oversees research subcontracts awarded to universities in the areas of Computer and Computational Science. Funding for these subcontracts can come from a variety of research budgets, almost all of which are managed by the Center for Applied Scientific Computing (CASC).

Visual Tracking of Multiple People

Michael C. Burl
University of Colorado, Boulder

Summary

Data mining has been defined as the process of extracting implicit, nontrivial, previously unknown, and potentially useful information from data in databases. Although much of the effort in the data-mining community has indeed focused on traditional databases, there is a growing interest in algorithms that can extract useful information from nontraditional data, such as video sequences. This trend is driven both by a heightened emphasis on security and by the ready availability of cheap hardware for quality video capture and analysis. It is well known that human observers have difficulty paying careful attention to massive amounts of data for any length of time. This problem is exacerbated if an operator is responsible for monitoring multiple video feeds simultaneously (e.g., a bank of display monitors). In addition, there are situations in which it is not feasible to have a human in the loop (e.g., space exploration, UAV surveillance, unattended sentinels that communicate with base only when something important happens).

In restricted domains where the environment can be controlled and the cost of errors is low or where there is a human in the loop, a few commercial video data-mining systems and advanced research prototypes that monitor people and their activities have begun to emerge. While these early systems provide an interesting degree of functionality, there is no completely satisfactory solution for the problem of autonomously monitoring activity in unstructured outdoor environments.

A significant portion of the research effort in mining video data is necessarily focused on the computer vision and image processing steps that must directly handle large volumes of raw data. The outputs from these specialized feature-extraction steps then serve as the input to higher-level data mining processes that answer questions relevant to a user. We focus on developing a prototype algorithm for extracting information from raw, surveillance-style video of an outdoor scene containing a mix of people, bicycles, and motorized vehicles.

A pilot study was conducted to develop techniques to extract useful information from image streams (video data), focusing primarily on the problem of tracking multiple people in video sequences taken from a distance. Reliable tracking is a prerequisite for supporting more sophisticated data-mining operations such as activity classification, anomaly detection, and vigilant monitoring for trigger events. A prototype algorithm, based on robust background estimation, spatial clustering, and multi-object tracking, was developed to process sequences of video frames into track sets. The extracted track sets encode the positions, velocities, and appearances of the various moving objects as a function of time. The prototype tracking algorithm was applied to a sequence of 18,000 gray-scale frames of an outdoor scene containing a mix of people, bicycles, and motorized service vehicles. The video frames were captured on the University of Colorado Boulder campus with a standard web camera

Continued

Michael C. Burl
University of Colorado, Boulder

Summary continued

situated approximately 25m above ground level. Results show that the algorithm is largely successful in tracking objects, but there are clear opportunities for further improvements.

Results also demonstrate the ability to perform data-mining queries over the extracted track sets. These preliminary studies show that useful information for surveillance/security operations or for input to public-planning and decision-making processes can be automatically obtained from raw video sequences even at the current level of tracking performance. Example utilities that that might be useful within this framework include setting up trigger events (e.g., detecting any vehicles entering a particular area, people exiting or entering a particular building), determining typical and anomalous patterns of activity, generating person-centric or object-centric views of an activity, classifying activities into named categories (e.g., walking, riding a bicycle), grouping activities into unnamed equivalence classes (clustering), and determining interactions between entities. We hope to pursue several of these directions in follow-on work.

Scientific Visualization for SAMRAI

Walter Freddy Herrera Jimenez
Oregon Graduate Institute

Summary

The goal of this project is to enhance the scientific visualization tool called VisIt to support H-Adaptive data generated by SAMRAI applications. Our enhancements are primarily concerned with supporting SAMRAI users' needs, but we intend to generalize our approaches, where possible, to support H-Adaptive data in general. H-Adaptive data is data in which the spacing between samples is varied (adapted) so that the mesh and variables may have higher fidelity (resolution) in localized areas. The kinds of work necessary in VisIt fall into roughly four categories: the GUI, the database, and I-blanking.

We worked as a group, including Peter Williams, Mark Miller, Hank Childs, and myself. My work was to develop the SAMRAI database reader and to modify AVT modules that allow VisIt to show default plots as soon as the user opens any SAMRAI database. The code was written in C++ and uses HDF5 and VTK libraries.

The current stage of the VisIt enhancement allows the user to open SAMRAI databases and use all the plots and operator tools to visualize and analyze its mesh and variables. By now, the visualization techniques consider that each level in the resolution hierarchy of the SAMRAI database is an independent mesh with its own variables. In the future, VisIt will have to "do the right thing" where multiple choices for what to render are available because of overlapping patches at multiple levels of resolution. In general, the right thing is for VisIt to display the highest resolution available in any one region of the mesh from the possible set of levels in the current selection.

Adaptivity and Related Algebraic Multigrid and Nonconforming Domain Decomposition Methods in a Massively Parallel Computing Environment

Raytcho Lazarov
&
Joseph Pasciak,
Texas A&M University

Summary

We have worked on several research projects on design, analysis, and implementation of parallel methods for various large finite-element problems. These include a least-squares finite-element method for Maxwell's equations, continuous and discontinuous finite-element approximations for second-order problems, and domain decomposition methods using matching and nonmatching grids (e.g., mortar approximations, penalty, and discontinuous Galerkin methods). These projects are related to the previous works and should be viewed as continuation and extension of the collaborative research between Texas A&M University and LLNL for the last five years. We have been working on discretization methods for partial differential equations (PDEs) that provide greater flexibility in the grid generation process, that increase the portability of various approximation methods and computer implementations, that enhance the capabilities of coarsening strategies in parallel algebraic multigrid methods, and that provide a posteriori error analysis for parallel adaptive methods. Our investigations produce competitive algorithms that can be used in various codes for complex applications in physics and engineering.

New techniques for Maxwell's equations. The goal of this project was to develop efficient algorithms for the numerical solution of problems arising from electromagnetic models. Specifically, we are interested in approximating Maxwell's equations and related eigenvalue problems. These are important in practical applications in which one needs to compute the electromagnetic field generated by prescribed current and charges or in the computation of the eigenmodes that will propagate through a given medium. The work was carried out by summer students T. Kolev and D. Copeland in collaboration with Joseph Pasciak from Texas A&M University and Panayot Vassilevski and Daniel White from Livermore Center for Advanced Scientific Computing (CASC).

One of the approaches considered is based on a weak variational formulation of div-curl systems corresponding to the electrostatic and magnetostatic problems. The finite dimensional approximation is a negative norm finite-element least-squares algorithm that uses different solution and test spaces. This algorithm allows for approximation of problems with low regularity where the solution is only in L^2 and the data resides in various dual spaces. The solution operators for the above problems are further used to obtain an approximation to the eigenvalue problem. The resulting discretization method has the advantages of avoiding potentials and the use of Nedelec spaces. A computer program was developed that applies the method to the Maxwell equations and the eigenvalue problem in the frequency domain. It is written in C++, in the framework of the AggieFEM finite element library, which supports complex geometries, local refinement, multigrid preconditioning, and OpenGL visualization. The code is based on solvers for the magnetostatic and electrostatic problems. It works on triangular, tetrahedral, and hexahedral meshes. It provides an eigenvalue solver, which allows for computations of blocks of eigenvalues and a solver for the full-time harmonic system.

Continued

Raytcho Lazarov
&
Joseph Pasciak,
Texas A&M University

Summary continued

A parallel version of this code was implemented based on MPI and CASC's Hypre preconditioning library.

A second part of the research involved the efficient computation of time-dependent Maxwell's problems. In this research, we considered a classical approach based on Nedelec curl-conforming spaces. The goal was to compare explicit time-stepping methods with implicit methods. The explicit method requires relatively little work per time step; however, the time-step size is constrained by a CFL-type condition. In contrast, the implicit method requires the solution of a system at each time step but is unconditionally stable so that larger time steps can be taken. We considered preconditioned iterative methods for solving the implicit problem with preconditioners based on algebraic multigrid. The computational results clearly showed that the implicit method with preconditioned iterative solve led to effective approximation at a significantly reduced cost on a variety of model problems.

Finally, we investigated iterative eigensolvers for the discrete equations that result from Maxwell eigenvalue problems discretized by Nedelec curl-conforming elements. The problem is to compute a modest number of the smallest nonzero eigenvalues, while avoiding the large null space of the curl–curl operator. We investigated a number of techniques for doing this. The most effective was a block iterative approach with preconditioning for the operator curl–curl plus the identity, while approximately enforcing the zero divergence condition. Initially, the zero divergence condition was approximately imposed by a preconditioned conjugate gradient iteration. Subsequently, this condition was imposed using the conjugate gradient iteration without preconditioning.

Domain Decomposition Methods for Mortar and Nonmortar Approximations. The interior penalty method aims at eliminating the need for additional (Lagrange multiplier or mortar) spaces and imposes (only approximately) the required continuity across the interfaces by an appropriate penalty term. In our approach, the jumps in the values of the functions along these interfaces are penalized in the variational formulation. For smooth solutions, we lose the optimal accuracy due to lower approximation at the interface, but we produce symmetric and positive definite discrete problems that have optimal condition number. Other features of our method are: natural parallelization of the iterative procedure and parallel mesh generation based on adaptive procedure that involves local-error estimators and indicators.

We also continued our research into using the mortar method for patching together nonconforming meshes that result when algebraic multigrid (AMG) methods are applied independently on subdomains distributed in a parallel computing environment. We have developed a code that implements the above technique using the AMGe spectral

Continued

Raytcho Lazarov
&
Joseph Pasciak,
Texas A&M University

Summary continued

agglomeration technique of Panayot Vassilevski. This code works for three-dimensional second-order model problems and develops the necessary parallel data structures for implementation of the global AMG preconditioner. Currently, the overhead associated with setting up the subprocessor agglomeration is the limiting factor in a moderately parallel application. To circumvent this problem, we propose to extend the approach by using other agglomeration techniques. We also propose to test the approach on machines with many more processors. A parallel code implementing the mortar method with algebraically constructed multiplier spaces was developed. The target application of this code is in the construction of parallel (including multigrid) preconditioners using element based (AMGe) coarsening in each subdomain. The work was carried out in close collaboration with our summer student T. Kolev from Texas A&M University and P. Vassilevski from CASC.

Discontinuous Finite-Element Approximation of Elliptic Problems. Jointly with P. Vassilevski we have proposed a framework for derivation of discontinuous finite-element discretizations of elliptic problems on matching and nonmatching grids and possibly many subdomains. The derivation of the schemes is based on penalty stabilization of the mixed formulation that allows elimination of the Lagrange multiplier. In particular, many known discontinuous finite-element Galerkin methods can be reproduced by the proposed framework. However, this could be considered as a way to develop approximations and domain decomposition methods for nonmatching grids. Stability, error analysis, and preconditioning methods for solving the discrete system of equations are developed. The results are in further development. We implemented the following three discontinuous Galerkin methods: the method of Baumann and Oden, the nonsymmetric interior penalty Galerkin (NIPG) method, and the interior penalty (IP) method.

The code was parallelized using MPI. All matrices are stored in Hypre ParCSR format, and the Hypre implementation of the iterative methods is used. The code has the following structure: (1) the root process reads an arbitrary tetrahedral mesh from a file and then refines it uniformly a given number of times; (2) the root process uses METIS to partition the new mesh into the number of subdomains equal to the number of processors used; (3) the root process sends to each processor its corresponding submesh (all elements in its subdomain), which will be the coarsest-level mesh for the multigrid; (4) the mesh is refined uniformly a given number of times in parallel (when multigrid preconditioner is used, at each level we assemble in parallel the matrix of the arising linear system, the interpolation operator, and the smoother used); and (5) on the finest level, the right-hand side is assembled and then the GMRES (Baumann–Oden and NIPG methods) or the PCG (for IP method) algorithm is used to solve the linear system iteratively.

Continued

Raytcho Lazarov
&
Joseph Pasciak,
Texas A&M University

Summary continued

We use a pure geometric multigrid method, with operators obtained by assembling at each level, thus generating (in general) nonnested bilinear forms. The natural embeddings of the discontinuous spaces were used as interpolation operators, thus giving rise to purely local interpolation–restriction. In the multigrid algorithm, the following two smoothers were considered:

• diagonal scaling—the preconditioner has elements on the main diagonal equal to the sum of the absolute weight of the rows. With this smoother, a variable V-cycle was used with three pre- and postsmoothing iterations on the finest level, and the number of smoothing iterations is multiplied by 4 from a level to the next coarser level;

• the ParaSails preconditioner from the Hypre library with the default settings. For the IP method, the SPD version of ParaSails was used. Here, V-cycle with one pre- and postsmoothing iteration was used.

Interaction with CASC members, visitors, and summer students. Raytcho Lazarov and Joseph Pasciak visited CASC for the month of August 2003. They were also in close contact with J. Jones, D. White, R. Falgout, C. Tong, and P. Vassilevski and interacted with other long- and short-term visitors R. Bank, T. Manteuffel, S. McCormick, C. Bacuta, and L. Zikatanov. Our scientific collaboration for this year resulted in joint papers that are submitted or already published. Three Ph.D. students from the program of computational mathematics at Texas A&M University, T. Kolev, V. Dobrev, and D. Copeland, spent three months each at CASC on a professional summer internship. All three Texas A&M Ph.D. students participated directly in the work of the contract. We plan to develop, implement, and test efficient parallelizable, scalable algorithms on various problems for treating both the geometrically refining spaces and algebraically coarsening grids. The future research should offer alternatives to the existing efforts in CASC for parallel AMGe method.

Enabling Large-Scale Data Access

Ling Liu
Georgia Institute of Technology

Summary

An ultimate goal of the Enabling Large-Scale Data Access project is to develop methodology and mechanisms for building a fully automated, end-to-end wrapper code generator. This can be achieved through the design, development, and integration of service-class descriptions with the XWRAP systems. The main idea is to provide mechanisms to enable XWRAP systems to take a generic description of a class of search interfaces (so-called service class) and the URL of a particular interface that is a member of the given class to produce a functional wrapper; this wrapper will take a class-specific query and produce an XML view of the query results obtained through this particular interface. In the second year of the Laboratory Directed Research and Development (LDRD) project, we focus on three main efforts: (1) developing concrete syntax for specifying the service-class descriptions, (2) designing and implementing a spider for surfing the Web and finding those Web sites that match with a given service class, and (3) integrating the service-class-based spider with the XWRAP system to automate the code-generation process of XWRAP.

We first give an overview of the work performed under the LDRD subcontract by the Distributed Data Intensive Laboratory (DiSL) at the College of Computing in Georgia Tech. We present the overview in terms of the deliverables defined in the statement of work.

Deliverable 1: *Java 1.3 code for a Web spider capable of taking a service-class description and a starting URL, crawling the Web, identifying interfaces that support a simple BLAST query, and generating a description of how to interact with that interface in the format required by the XWRAP Composer program described* in Deliverable *(3).*

Deliverable 2: *An XML service-class description for a BLAST query.*

Deliverable 3: *A JAR file containing a demo version of the wrapper program generated by XWRAP Composer, which is capable of taking the service-class description generated by the spider referenced in Deliverable (1) and producing a java wrapper for that source.*

Deliverable 4: *A short whitepaper describing the interface description format output by the spider in Deliverable (1) and used by XWRAP Composer (3).*

Deliverable 5: *A final report describing the work completed during this subcontract, any problems that occurred, possible extensions to the system, and how the software can be installed and used.*

We have completed all the tasks described in the above deliverables and report briefly selectively here.

Service-Class Description and Source Discovery. One of the main results of our second-year effort is the development of the service-class descriptions for a selection of Bioinformatics Web sources and the service-class-driven spider that is capable of

Continued

Ling Liu
Georgia Institute of Technology

Summary continued

crawling the deep Web and discovering the sources that match the given service-class description.

Concretely, our approach to discovery and classification of Web sources groups them into *service classes* that share common functionality but not necessarily a common interface. Service classes are specified by a *service-class description*, which uses an XML format to define the relevant aspects of a category of Web sources from an application's perspective. The service-class-description format supports the source-discovery problem by providing a general description of the type of source that is considered interesting. It defines the data types that comprise the service arguments as well as any intermediate types that may appear in a source. It establishes a general description of the interface used by source-class members and outlines intervening control points. Finally, it lists examples that are employed during source evaluation. Each of these components is described in detail in the remainder of this section.

The first component of a service-class description specifies the data types that are used by members of the service class. Types in this context are analogous to those in programming languages. They are used to describe the input and output parameters of a service-class and any data elements that may be required during the course of interacting with a source. The control flow graph consists of a set of states connected by edges. Each state has an associated type; data from a Web source is compared against the type associated with the control-flow states to determine the flow of execution of a source from one state to another.

The service-class description examples provide the mechanism by which a source can be analyzed and tested for service-class membership. Examples are tied to one or more paths through the control flow and specify data for the input types at a particular control-flow state; there may be many examples specified in a service-class description. Each example consists of a set of arguments that are paired with form parameters on the Web source. An argument has a name, a data type defined in the types section, a set of hints, and a data value. Hints leverage the observation that because Web pages are constructed by humans, form parameter names and values tend to reflect their purpose. The hints provide clues that allow the source analyzer to tie an argument to form parameters with a better chance of picking the correct parameter.

Source Analyzer. We have constructed the code to accept a service-class description as input and to use that description to determine service-class membership for a given Web source. Source analysis encompasses the following steps:

- input parsing an object materialization,
- test-query generation, and
- query execution and response analysis

Continued

Ling Liu
Georgia Institute of Technology

Summary continued

During input parsing, the source analyzer reads in the service-class description and constructs a set of objects that represent that description and provide useful methods for operating on data obtained from a Web source. Our work here was to construct the objects that represent the service-class in memory along with the desired functionality for those objects. The mechanics of parsing a service-class description into in-memory objects are handled by the Apache Digester library, which takes an XML file and a set of rules and generates arbitrary Java objects according to the specifications in the rules. The source analyzer also retrieves the start page of the supplied Web source during input parsing. Network communications and HTML handling are provided by the HTTP Unit programmatic-user agent library.

Once the input data has been processed, the source analyzer uses the examples in the service-class description and the forms from the Web source's start page to generate a set of queries that will be used to probe the source. The analyzer's goal is to discover the function of the various inputs on the form and thereby determine if the source matches the service-class description. Queries are generated by computing all possible permutations of the example arguments with the form parameters. From this set, queries are chosen to be executed in an order determined by a scoring function that gives higher priority to queries where the form parameter chosen for a particular argument matches the hints specified by that argument. In this way, the source analyzer tries to execute queries that it deems more likely to succeed first.

After generating the list of queries to execute, the source analyzer begins probing the site and testing the responses. Response analysis is handled by the type library. A Web source's response is tested by the type library using the type defined for the current control state in the service-class description. If the response matches the specified type, the analyzer progresses to the next control state; if the current control state is an end state, the Web source is declared to be a match for the service-class description and processing concludes.

XWRAP Composer. XWRAP systems aim at semi-automatically generating Java code (Wrapper) for extracting useful content from Web pages. XWRAP Elite is designed to generate wrapper programs for extracting information (primarily query-answers) from the deep Web data sources. XWRAP Composer is designed on top of XWRAP Elite, aiming at providing information extraction across multiple linked Web pages. A typical example is the Matt's scenario [3].

The XWRAP Composer requires three types of input descriptions to generate the Java code of the given Web source from the service-class specification.

- The first input description is called Interface Description, which is used to specify the URL of the Web source to be wrapped and the keyword or entry query used to obtain the pages from the search interface of the given Web source.

Continued

Ling Liu
Georgia Institute of Technology

Summary continued

- The second input description is called Output Service-class Description. It describes the XML tagging information and how extracted data content can be tagged as specified.
- The third input parameter is the Composer Script Language, which consists of two main components: the QA control logic specification and the data extraction logic required. The QA control logic specifies the possible query-answer control flows of the search interface of a given Web site. An example of such control logic is as follows. The NCBI BLAST interface will respond to a BLAST query with at least three different control flows:
  - The first one is the waiting page, which tells how long one needs to wait for the results to be returned.
  - The second one is the full result page, which is returned by following the indirection link on the intermediate waiting page.
  - The third one is the error page, which is returned when following the indirection link on the waiting page.

Thus the Composer wrapper needs to generate code pieces to handle all three cases.

Generating a script definition for XWRAP Composer from the source analyzer involved the creation of code to further analyze a discovered Web source and to tie the various pieces of information obtained from the service-class description and query probing into a form that Composer could understand. Additional site analysis was required in order to determine fully the capabilities of a recognized Web source. For example, in BLAST, it is possible for a single source to provide a gateway into many databases, each of which should be available in the system and thus require wrappers. The additional site analysis is similar to the site analysis used when initially testing the source, but involves more exploration of the various options presented by the Web source's forms.

Once the analyzer has obtained all of the necessary information, script generation can begin. Each script is divided into several tasks. The source analyzer begins by constructing a task that will read in the service-class description to provide later tasks with type definition and other data they will need to complete their work. The second task created by the source analyzer takes input from a user in the system and constructs the messages needed to submit the query to the Web source. Next, the script specifies a task that will annotate the Web source's results with XML tags. The final task takes the annotated results and constructs data objects that will eventually be passed to the user.

The generated XWrap Composer script follows the interface specification laid out in the attached specification document and must conform to the schema definition. Two versions of the schema definition are provided: one in RelaxNG compact syntax for easier readability and a standard XSD description generated from the compact syntax. For tools to make the translation, see http://thaiopensource.com/relaxng/.

Continued

Ling Liu
Georgia Institute of Technology

Summary continued

Results Obtained from Integration Efforts. One of the main objectives of Deliverable (3) is to produce a design document describing how the service-class description of a remote data source generated by the spider referenced in Deliverable (1) is fed into the XWRAP Composer to generate wrappers in Java for that source without any human involvement. A demo is included to show the feasibility of such design.

There are two challenges for providing a fully automated, end-to-end wrapper generator system. First, we need mechanisms that can provide all the information required by XWRAP code-generator system through human interaction with XWRAP. To our knowledge, all wrapper-generator systems to date are semi-automated, and they require wrapper developers to enter information to the code-generator systems at different stage of the code-generation process.

In XWRAP. there are three manual steps that require human input in its code-generation process.

- First, it requires a manual input of the URL of the Web site to be wrapped.
- Second, it requires manually writing the Composer script to capture the control logic of the query-answering patterns of the Web site to be wrapped.
- Third, it requires manually inputting the data-extraction complications such as element alignments, or multi-page extraction through links.

The second year of the LDRD project focuses on mechanisms that can provide full automation by utilizing the manually developed service-class description to capture all the human-input information required by the XWRAP code generator. This effort is carried out in three steps. First, we manually create the service-class description tailored to the type of Web sites to be wrapped such as BLAST sites. Then, we find the Web sites matching a given service-class description. Finally, we integrate the service-class description with the XWRAP Composer, such that the service-class description for XWRAP Composer will provide interface, outerface description as well as the QA control logic and the data-extraction logic required by the XWRAP Composer code generator.

Concretely, the service-class-description-based spider is used to discover a set of Web sites that share a similar search interface such as BLAST. Thus, instead of entering a specific URL to the XWRAP Composer, one can simply request for a wrapper to be generated capable of providing some specific services such as BLAST. The service-class description is used to find the concrete Web site offering the type of services — for example, the BLAST sites. For each of such Web sites, its URL can be passed to the XWRAP for generating a wrapper for that site.

Continued

Ling Liu
Georgia Institute of Technology

Summary continued

It is observed that many Web sites share similar search interfaces and QA control logics [2]. Therefore, by including the manual specification of the QA control logic required by XWRAP to wrap a Web site in its matching service-class description, it makes it possible to automate the generation of the QA control-logic part of the XWRAP Composer script.

For the data-extraction logic, Web sites offering the same types of services tend to differ in how they organize and lay out the presentation of the content. Therefore, by moving some type of data-extraction logic into the service-class description, a certain number of Web sites that provide the same type of services will not be discovered due to the mismatch in terms of their data-extraction logic.

Our experience with the integration efforts yields two observations:

(a) the types of BLAST sites where service classes work well may cause the object-extraction algorithms used in Omini to fail; thus, XWRAP will not be able to generate wrappers, and
(b) Web sites that XWRAP/Omini can work well may not be the best Web sites for the service-class spider (such sites will need more complex interactions between wrapper developer and the XWRAP system).

One possible solution is to add what XWRAP needed to generate the Entrez wrapper (the alignment rules) to our Entrez service-class specification. This will allow us to integrate the two pieces of code, which takes the input to the service-class spider and returns an Entrez wrapper. Another solution is to find those Web sites that a service-class generator can work more or less correctly, and Omini will also be able to find the correct objects.

There are three papers directly published in this project.

- The first paper [1] reports our work on source discovery and classification. The claimed contribution of the paper is the use of an abstract description format for the classification of arbitrary Web sources. The paper describes the service-class description, the source analyzer, and the validation results obtained by testing the code against a variety of Web sources. Using BLAST as a case study, we showed that the source analyzer is able to correctly identify 66% of the supplied BLAST sources with no false positive identification of non-BLAST sites. The paper was accepted to the Atlanta Bioinformatics conference and is scheduled for publication in November 2003.
- The second paper reports our new data extraction algorithms [2] which appeared in the IEEE International Conference on Data Engineering with *a* 14% acceptance rate. This paper presents QA algorithm for focused extraction.
- The third paper [3] summarizes the issues involved in data integration over more than 500 Web sources.

Continued

Ling Liu
Georgia Institute of Technology

Summary continued

Publications

[1] Daniel Rocco and Terence Critchlow. *Automatic Discovery and Classification of Bioinformatics Web Sources.* In Proc. Atlanta Bioinformatics Conference. 2003 (to appear)

[2] James B. Caverless, Ling Liu, and David Buttler. *Probe, Cluster, Discover: Focused Extraction of QA-Pagelet in Deep Web Data Sources*, To appear in IEEE International Conference on Data Engineering. 2004.

[3] David Buttler, Matthew Coleman, Terence Critchlow, Renato Fileto, Wei Han, Calton Pu, Daniel Rocco, Li Xiong. *Querying Multiple Bioinformatics Information Sources: Can Semantic Web Research Help?* SIGMOD Record, Vol. 31, No. 4. 2002 (December)

Optimizing a Parallel Code for the Simulation of Neuronal Networks

Peter Pacheco
University of San Francisco

ISCR Institute for Scientific Computing Research

Summary

Object-oriented NeuroSys is a collection of parallel C/MPJ programs for simulating very large networks of biologically accurate neurons. It includes two principal programs: ooNeuroSys, a parallel program for solving the large systems of ordinary differential equations (ODEs) arising from modeling the interconnected neurons, and Neurondiz, a parallel program for visualizing the results of ooNeuroSys. Both programs are designed to be run on clusters and use the MPJ library to obtain parallelism.

NeuroSys was originally developed in the late 1990s by the University of San Francisco's Applied Mathematics Research Laboratory. It obtained parallel efficiencies of better than 90% on networks of 250,000 Hodgkin-Huxley type neurons and a fast-ethernet-connected cluster of 32 Intel processors running Linux. However, its design is essentially structured, and as it grew, it became very difficult to incorporate new features and to improve existing ones. So in the summer of 2001, we began a complete rebuilding of NeuroSys. One of the central features of the new system is an object-oriented design that makes maintenance relatively easy and actually improves both performance and scalability. Because of the relative ease with which C codes can be optimized, we chose to write ooNeuroSys in C. Neurondiz is divided into a computational engine or "backend" and a "frontend" for managing the display. The backend is also written in C, and there are two versions of the frontend: one written in Java for maximum portability; the other written in C++ using the Qt and OpenGL libraries for maximum performance.

During October 2002–May 2003, we worked on a number of approaches to improve the performance and usability of ooNeuroSys. Perhaps the most significant improvement in the overall performance for ooNeuroSys was obtained by replacing fairly simple, nonscalable parallel ODE solvers with parallel CVODE, a code developed by researchers in the Center for Applied Scientific Computing (CASC). Another significant enhancement came in the development of adaptive methods for structuring the interprocess communication. A third avenue to performance improvement was obtained by selectively exposing incomplete data structures and replacing function calls —especially accessor functions — by macros. A fourth avenue for performance enhancement has come from the improvement of the I/O schemes used in the program.

From a computer scientist's point of view, one of the more interesting aspects of ooNeuroSys is that the interconnection networks in the systems it models range from being very sparse to very dense. Thus, it is impossible to know the best algorithm for interprocess communication until runtime. In order to address this, we first developed several different algorithms for interprocess communication. However, we found that only two sufficed for the best performance: one for densely interconnected networks and one for sparsely interconnected networks. At runtime we use a heuristic scheme to

Continued

Peter Pacheco
University of San Francisco

Summary continued

evaluate the density of the interconnection network. If the interconnect lies near the "crossover point" from sparse to dense, we run a small benchmark of etch scheme to empirically determine which interconfleet is superior. Since the interconnects change infrequently if at all, this approach adds very little to the overall cost of running the heuristic, and the benchmark is more than made up for by the overall reduction in the cost of communication.

Although much work has been done in developing sophisticated software for interfacing to *parallel* filesystems, little has been done to improve the performance of parallel software for programs that interface to conventional Unix I/O. This is an important problem since many, if not most, Linux clusters use multiprocessor nodes with dedicated local disks, and most of these systems simply use the ext2 or ext3 Linux filesystem. Thus, one of the main foci of our work has been optimizing the I/O performance of multiprocessor, especially dual-processor, Linux systems running parallel programs. Our research has shown that in order to optimize I/O-intensive programs running on such systems, it is necessary to use threads. However, for programs that make less intensive use of the I/O subsystem, we have found that a pure MPI approach combined with the use of the mmap() and msync() system calls provides excellent overall performance—performance as good as, or even somewhat better than, a mixed MPI/Pthreads approach.

In order to improve the usability of ooNeuroSys, we added a Python interface. This addresses one of the most challenging problems we faced in the original version: the difficulty of converting a system of ODEs modeling a neuron into a collection of C functions with the proper interface to the rest of the program. The Python interface is much simpler (it requires only a nodding acquaintance with programming) and because Python is interpreted, users get almost instant feedback on both the correctness and the usefulness of their model code.

We developed the frontend-backend design of Neurondiz because we assumed that our users would mainly have access to computing systems with two basic configurations. The first configuration assumes the user has access to a parallel computer with essentially no graphics capability. For example, a user who accesses a remote parallel system at a supercomputer center via the internet would fall into this category. The second configuration assumes the user possesses a parallel system in which one of the compute nodes is directly connected to the display. For example, a user with a small cluster might use this setup. For users with the first setup, the backend, the compute-engine, communicates with the frontend, the display management software using sockets. For users with the second setup, the frontend and backend are, effectively, part of the same program and they communicate via function calls. Currently our Java frontend supports sockets for communicating with the backend, and our C++ frontend

Continued

Peter Pacheco
University of San Francisco

Summary continued

supports direct function calls. We plan to ultimately implement both designs with both languages, using JNI for the Java frontend.

During the period of the LLNL visit, our main focus with Neurondiz has been on adding features to the user interface. The original version simply opened a window and stepped through the results generated by the ODE solver. It displayed a rectangular grid of disks — one disk for each neuron — and through color changes in the disks showed the changes in neuron membrane voltages. We have added the capability to "blow-up" a subset of the display for more detailed information. The new version also allows a user to select an individual neuron and display a membrane voltage vs. time plot for the neuron.

The neuroscientists we worked with asked us to provide information on the synaptic interconnections between the neurons. After some discussion we added two functionalities. The first allows the user to select a subset of neurons and display the synaptic interconnections among the selected neurons. The second allows the user to select a source and a destination neuron and a bound on the number of synaptic connections. The software then finds the subnetwork consisting of all paths of synaptic connections from the source to the destination that have length less than or equal to the bound. Both functionalities make use of the graph visualization software package, Graphviz.

For the near future, we plan to fully integrate all the Neurondiz functionalities into both the Java and C++ frontends. We also plan to further optimize the I/O and communication of ooNeuroSys. For the longer term, we plan to integrate the use of graph partitioning software into ooNeuroSys and Neurondiz.

The work done at LLNL has provided tremendous improvements in both the performance and usability of Object-oriented NeuroSys. We expect that computational neuroscientists will benefit greatly from these improvements. We also expect that the wider community of users of parallel scientific computing will benefit from much of this software and many of its design features.

Complex Networks

Alex Pothen
Old Dominion University

Summary

Graphs in which the average distance between a pair of vertices is small, i.e., O(log |*V*|), where |*V*| is the number of vertices, are called *small-world* graphs or networks. Such networks are of interest currently since they arise in social, biological, and web-community networks. Small-world graphs can be classified into several classes: Erdos-Renyi random graphs, linearized chord diagram (LCD) random graphs, and the Strogatz-Watts-Kleinberg (SWK) class of graphs.

In Erdos-Renyi random graphs, each edge is present with a probability *p(n)*, where *n* is the number of vertices. These graphs have been well studied in the literature, e.g., Bollobas' book on Random Graphs. It is well known that these graphs cannot be partitioned well in the sense that they cannot be partitioned into subgraphs with approximately equal numbers of vertices such that few edges join the different subgraphs.

LCD graphs have been studied by Barabasi and others, and we follow the construction of Bollobas. We build the graph through a random process, beginning with, say, a clique on three vertices. At each stage, we add a new vertex, joining it to two vertices in the current graph, with the probability of the endpoint proportional to its current degree. Thus, it is more probable that a new edge will join the new vertex to a previous vertex of high degree, than to a vertex of low degree. The LCD model leads to graphs whose vertex degrees satisfy a power-law, i.e., $N(d) = Ad^{-B}$, where *N(d)* is the number of vertices of degree *d*, and *A* and *B* are constants. At the beginning of this project, we had hoped that it might be possible to partition such graphs by removing the few vertices of high degree from the graph initially, partitioning the residual graph, and then adding the removed vertices to appropriate subgraphs. We implemented such an approach, but it did not give good partitions: a constant fraction of the edges in the graph were cut by the partition. In hindsight, this result was to be expected. Recently, Bollobas and Riordan have shown that the LCD model generates graphs with almost the same properties as the Erdos-Renyi random graph model. Hence it should be possible to prove that the LCD model leads to graphs that do not have good partitions similar to the ER random graphs.

The SWK graphs begin with a rectangular grid graph (in two dimensions) or a cubic grid graph (in three dimensions), with each vertex joined by edges to its neighbors in the north, south, east, and west in the two-dimensional grid (and similarly for the 3D grid). We now add a set of random edges, by choosing the endpoints at random. If we choose the pairs of endpoints without replacement, then at most one random edge can be incident on each vertex in the graph. Assume that we wish to partition the graph into *P* subgraphs. Then the probability that the random edge joins two vertices in the same subgraph is *1/P*, and that it will be cut by a partition is *(P - 1)/P* . Hence we expect that most of the random edges will be cut. We classify the random edges into two classes: *local* edges that join

Continued

Alex Pothen
Old Dominion University

Summary continued

vertices that are close to each other in the grid, and *remote* edges that join vertices that are distant in the grid.

To make this classification precise, we need to define the concept of distance in SWK graphs. The usual definition of the distance is the number of edges in a shortest path between two vertices in a graph. We define a *grid distance* of two vertices in SWK graphs as the number of edges in a shortest path that includes only edges from the grid and none from the random edges. Thus, the two endpoints of a random edge could have a large grid distance between them if the random edge is a remote edge.

We now describe an algorithm that has proved to be successful for computing a good partition for SWK graphs. We do not use information about which edges are original grid edges, and which have been randomly added, in the algorithm. The idea is to seek to identify the remote edges, remove them from the graph, and then to partition the residual graph. We identify a remote edge by temporarily removing the edge, computing the distance between the two endpoints of the edge, and classifying it as remote if it is larger than a threshold value. There is no guarantee that the distances obtained in this manner correspond to the grid distance (as we have defined it earlier), because random edges could be included in the paths we consider. But since random edges are chosen without replacement of endpoints, at most one random edge can be incident on each vertex in the graph, and with high probability we identify most of the remote edges. It is also clear that local edges will be correctly classified in this approach. (Even if we add random edges with replacement of endpoints, the probability that two or more random edges are incident on the same vertex is low, and this approach works in practice.)

The algorithm for partitioning an SWK graph then is to consider each edge in turn, compute the distance between its end points using paths that do not include that edge, and then temporarily remove the edge if it is classified as remote. After removing edges that have been classified as remote, we then partition the residual graph, using METIS, one of the well-known graph partitioning packages. Computing the distance between the endpoints of an edge can be done in $O(|E|)$ time, where $|E|$ is the number of edges, using a breadth-first-search (bfs). The algorithm can be implemented in $O(|E|^2)$ time, since $|E|$ edges need to be classified as remote or local. In practice, of course, a bfs can be terminated as soon as we reach the second end point of an edge from the first end point, and hence the algorithm runs quite fast relative to the partitioner.

We have generated a test graph consisting of a 100x100 grid, with a five-point stencil (each vertex connected to its four neighbors on the grid), and then added, in 99 steps, 100 random edges in each step. In the following two figures, we show the result of partitioning the final graph with 9900 random edges added, first using METIS directly, and then using the new algorithm that identifies and removes the remote edges before partitioning it

Continued

Alex Pothen
Old Dominion University

Summary continued

with METIS. The figures show quite clearly that the random edges confuse the multilevel partitioner METIS, which then computes a partition into two sets that loses the notion of locality in the grid graph. On the other hand, once most of the remote random edges have been identified and removed, then the partitioner partitions the grid graph into two subgraphs with good locality properties.

We believe that these results are of interest to community networks, where a few individuals serve to reduce the distance between two sub-communities that otherwise have little overlap. Results in the random graph literature suggest that a few such edges reduce the diameter of random graphs to O(log |V |). Hence the partitioning algorithm described here should be useful to partition large-scale community graphs for distributed computations.

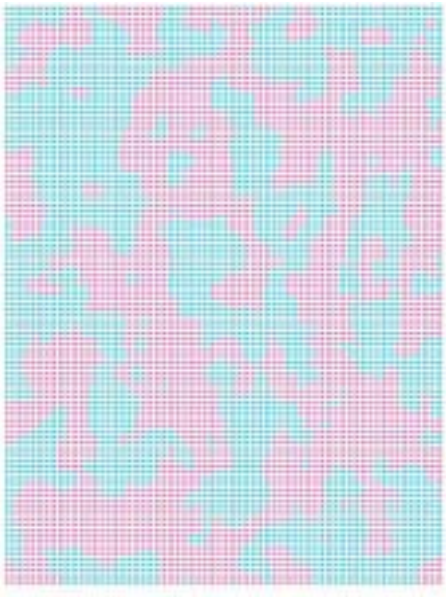

Fig. 1. The 100x100 (five-point) grid with 9900 edges added at random partitioned with METIS.

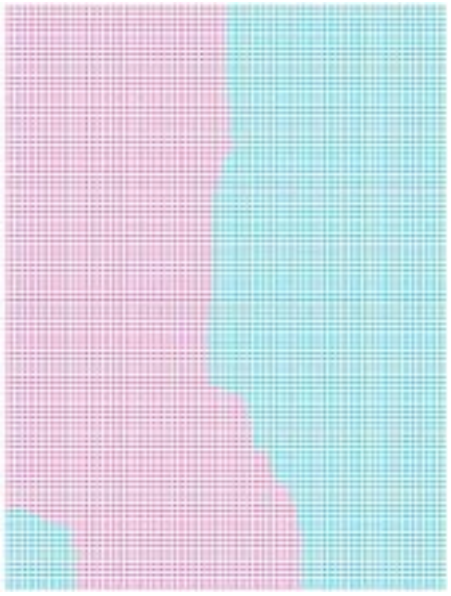

Fig. 2: The 100 x 100 (five-point) grid with 9900 edges added at random partitioned, with the new algorithm. The algorithm first identifies remote edges, removes them from the graph, and then uses METIS to partition the residual graph.

FOSPACK

John W. Ruge
Front Range Associates

Summary

The goal of this subcontract is continued development of the FOSPACK code, which is a package developed by John Ruge for the automatic discretization and solution of First-Order System Least-Squares (FOSLS) formulations of partial differential equations. Both 2D and 3D versions have been written, although the 3D version is more rudimentary. FOSPACK takes a user-specified mesh (which in 2D can be an unstructured combination of triangular and quadrilateral elements) and specification of the first-order system, and produces the discretizations needed for solution. Generally, all specifications are contained in data files, so no recompilation is necessary when changing domains, mesh sizes, problems, etc. Much of the work in FOSPACK has gone into an interpreter that allows for simple, intuitive specification of the equations. The interpreter reads the equations, processes them, and stores them as instruction lists needed to apply the operators involved to finite element basis functions, allowing assembly of the discrete system. Quite complex equations may be specified, including variable coefficients, user defined functions, and vector notation. The first-order systems may be nonlinear, with linearizations either performed automatically or specified in a convenient way by the user. The program also includes global/local refinement capability. Solution of the linear systems is performed with an algebraic multigrid solver, which is very well suited for such problems.

The specific tasks to be performed were as follows:

1. Continue work on both 2D and 3D FOSPACK codes to improve modularity of the program and generality of meshes allowed.

2. Continue work for allowing higher-order elements, with the ultimate goal of a full h-p refinement algorithm.

3. Implement and test methods for dealing with time dependence.

4. Implement and test methods for incorporating added algebraic constraints (e.g., slide surfaces).

5. Work toward parallelization of the FOSPACK code.

6. Work on development and incorporation into FOSPACK of alternative solution methods, particularly self-correcting Algebraic Multigrid [AMG] and smoothed aggregation.

7. Continue exploration of the use of FOSLS in problems of interest to LLNL personnel.

Continued

John W. Ruge
Front Range Associates

Summary continued

FOSPACK was originally written as a stand-alone code, meant for easy testing and solution of FOSLS formulations. While it was somewhat modular (with clear interfaces between the four main tasks: equation interpretation, domain/mesh definition, assembly of the linear systems, and solution of linear systems), it was never meant to be either fully modular or a library of routines with clearly defined interfaces. However, several factors make improving modularity an important goal. The first is simply that the amount of planned work is too great to be accomplished practically by one person. Efficient division of labor requires a clearer separation of various code parts, along with clearly specified interfaces. The second factor is that, as the code develops, the application to a wider range of problems, and more realistic problems, becomes necessary. The capabilities of some parts of the code, particularly the mesh generation, are not sufficient for such generality. Use of existing packages requires a clearly defined interface suitable for inclusion of such packages. The use of alternate solvers, discussed later, also requires a suitable interface.

Issues concerning these two factors are being addressed separately as they arise. Much of the emphasis on division of labor concerns the introduction of higher-order elements, discussed in more detail below. While use of third-party software for such elements was considered, there seemed to be no clear way to separate the definition of finite element space from matrix assembly. Instead, a number of tasks were defined that were required in matrix assembly, functional computation, etc., with interfaces specified. This allowed the matrix assembly and functional computation routines to be rewritten with no dependence on the function space chosen and knowledge of the spaces contained only in this new set of routines.

For mesh generation, a number of packages were examined, and a number of issues were considered, including generation of the initial mesh, refinement, and derefinement. It was decided that the best approach was to use (and enhance) the refinement and derefinement capabilities already in FOSPACK, while allowing for an initial mesh produced by external mesh generation packages. Such a capability is already partly included, and a fairly clear interface exists. Some restructuring of the code was performed, allowing for reading a “raw” mesh (list of nodes and elements along with an element-node correspondence table). Additional information (such as boundaries) can be specified in separate files, and conversion into the form required by FOSPACK is also performed separately.

Much of the work during this performance period was directed toward the introduction of higher-order elements. Currently, this is included in the 2D version only, although the routines developed were structured in such a way as to allow fairly straightforward extension to the 3D case. Higher-order functions are restricted to quadrilateral (not triangular) elements and can essentially be any desired order (although there are

Continued

John W. Ruge
Front Range Associates

Summary continued

practical limitations, such as storage and the slower convergence behavior of the AMG solver as the order is increased). The coarsest mesh problem is defined using bilinear elements, and higher-order functions are introduced as a refinement level (in a process called "p-refinement"). Initially, only global p-refinement was allowed, and only on a mesh without local h-refinement (where h-refinement is the usual element subdivision). Work is just concluding allowing for arbitrary p- and h-refinement (where the determination of p- or h-refinement can be made separately for each element). The next stage is to determine which refinement strategies are beneficial or practical.

Time stepping has been added to both the 2D and 3D versions of FOSPACK. Currently, this is implemented by explicitly writing the time-dependent terms in the first-order system in difference form (e.g., $(u-u_{old})/dt$), where u_{old} is the saved solution from a previous time step or the user-supplied initial condition. For this, a new function called *prev* was introduced, so that the time-dependent term could simply be written as *(u-prev(u))/dt*. FOSPACK then knows to allocate space and store the values of *u* at the end of each time step. FOSLS is then applied to this system, resulting in backward Euler time stepping. The time step *dt* used can be specified in several ways and can either be fixed throughout the solution process or can be adjusted by the user based on the evolving solution. (The latter requires some user coding.)

The addition of algebraic constraints has not yet been fully implemented, although little additional coding is needed. A nice method allowing the user to specify such constraints is needed.

Work towards parallelization of the FOSPACK code is continuing. As a first step, the existing AMG solver has been replaced with a call to BoomerAMG through the HYPRE interface. Comparisons with the serial AMG solver are currently in progress. In addition, the matrix assembly and functional computation are being studied for parallelization to determine whether existing code can be easily modified, or whether extensive rewriting is necessary.

Development of self-correcting AMG and smoothed aggregation continues. Tests show that both methods can work well for a number of problems for which AMG is not currently well suited. At the moment, only fairly simple model problems have been studied, and the methods have not yet been implemented in FOSPACK.

Much has been accomplished in this six-month project, with the main advances the introduction of higher-order finite-element spaces and time-stepping capability. While more needs to be done here, this already allows much increased accuracy for invested work and the study of FOSLS methods applied to a wider range of problems.

Adaptive Numerical Methods for Reactive and Nonreactive Flow on Overlapping Grids

Donald W. Schwendeman
Rensselaer Polytechnic Institute

Summary

We develop an adaptive numerical method for the accurate calculation of high-speed reactive and nonreactive flows on overlapping grids. In the reactive case, the flow is modeled by the reactive Euler equations with an assumed equation of state and with various reaction rate models. The nonlinear hyperbolic partial differential equations are solved with an unsplit, shock-capturing scheme; a Godunov-type scheme computes fluxes and a Runge-Kutta error control scheme computes the source term modeling the chemical reactions. An adaptive-mesh-refinement (AMR) scheme has been implemented in order to locally improve grid resolution. The method may also be applied to nonreactive flow problems, in which case the source terms are simply set to zero. The code uses composite overlapping grids to handle complex flow geometries. It is part of the Overture-OverBlown framework of object-oriented codes, developed in collaboration with Bill Henshaw, David Brown, and other members of the Overture team within the Center for Advanced Scientific Computing (CASC). During FY 2003, we explored two-dimensional detonation-propagation problems using an ignition and growth reaction model with a mixture JWL equation of state, extended the method with AMR to handle three-dimensional reactive and nonreactive flows, and also extended the method to handle two-dimensional moving meshes.

An ignition and growth reaction model is one of several reaction models now available in the reactive Euler code. This reaction model is designed to handle heterogeneous explosives and assumes a two-component mixture consisting of a (solid) reactant and a (gaseous) product. Each component is assigned a JWL-type equation of state. The reaction rate for the ignition and growth model involves several steps, each with its

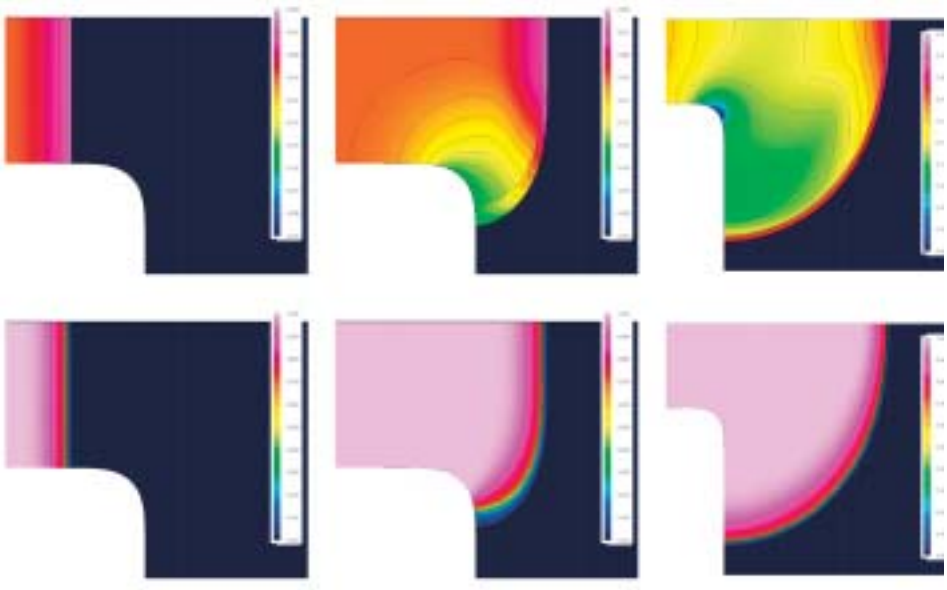

Fig. 1: Corner turning for an ignition and growth reaction model. Top row is pressure and bottom row is reaction progress.

Continued

Donald W. Schwendeman
Rensselaer Polytechnic Institute

Summary continued

own rate, typically pressure-dependent, and its own depletion law. It was used to study various problems involving detonation diffraction. One such problem involves detonation corner turning, as illustrated in Figure 1. In this problem a steady Chapman-Jouget detonation is propagating in a solid explosive (from left to right in the figure). The parameters used in the ignition and growth model are those for PBX9502. Of particular interest in this problem is the behavior of the detonation as a result of the interaction with the corner, and whether the detonation would be weakened to the point of failure. This calculation uses a composite overlapping grid with approximately 40,000 grid cells on the base level and two additional levels of AMR grids with a refinement factor of 4, so that the reaction zone is well resolved on the finest grid level. The corner is replaced by a 90° circular arc, whose radius is of the order of the length of the steady CJ reaction zone. Thus, the corner remains sharp on the length scale of the flow, but the corner singularity has been removed. As the detonation turns the bend its leading shock weakens and the reaction zone behind it lengthens. The strength of the shock, however, never falls below the level of compression needed to maintain the ignition step of the reaction model, and thus the reaction does not fail. In fact, the calculations show that the detonation ultimately strengthens once it has completed its interaction with the bend and is propagating down the straight vertical wall.

An extension to handle three-dimensional reactive and nonreactive flows is currently under testing for both accuracy and efficiency. One such test involves shock diffraction by a sphere in a nonreactive gas, as illustrated in Figure 2. In this calculation, a planar shock propagates from left to right in a channel with rectangular cross section and is diffracted by solid sphere centered at the origin. The base grid consists of a Cartesian grid for the channel and two orthographic grids to handle the boundary of the sphere. One AMR level with refinement factor equal to 4 is used in order to locally increase the grid resolution. The outline of the AMR grids may be seen along with contour plots on perpendicular cut planes for the time t = 0.6 in Figure 2. At this time, the planar shock has just impacted the sphere, and a high compression region and a reflected shock have developed in front of the sphere. At later times, there are too many AMR grids to be shown clearly in the figure so that only the contours of density are shown. The solution is well represented on the grid and the contours of density show the expected behavior of the diffracted

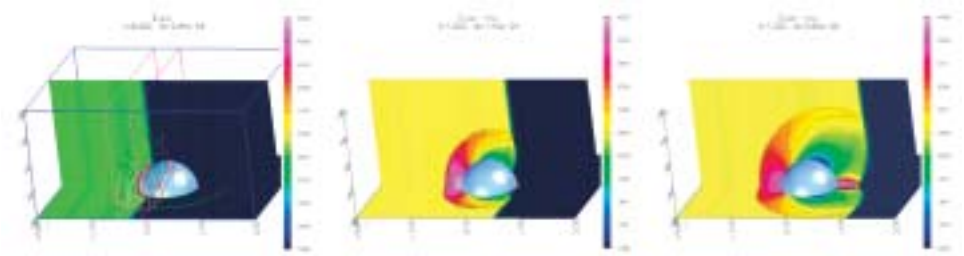

Fig. 2: Planar shock diffraction by a sphere. Density at t = 0.6, 1.0 and 1.3.

Continued

Donald W. Schwendeman
Rensselaer Polytechnic Institute

shock system. The contours of density also indicate that the numerical solution maintains axial symmetry even though this symmetry is not assumed in the fully three-dimensional calculation. This solution gives an indication of the problems being considered for testing. Other flows have been computed using the three-dimensional version of the code and have been tested for accuracy, but more testing is still needed, and some areas in which the code could be made more efficient have been identified, in particular the overhead needed to manage the AMR grids.

Work has begun to extend the code to handle discretizations involving moving meshes. The main issue here centers on the incorporation of a grid velocity into the numerical scheme. This involves modifications of both the calculation of the Godunov flux and the calculation of the source. The appropriate modifications to the flow-solver have been made and the work will now focus on testing the numerical method. One of our first tests is shown in Figure 3. In this test, a two-dimensional, nonreactive gas is at rest in a square channel and is disturbed at time *t=0* by the impulsive motion of an imbedded cylinder. The cylinder moves with a constant velocity from right to left. The motion of the cylinder is handled by the moving annular grid shown in the figure. The corresponding contours of density for the flow are shown above the grid. As the time-stepping proceeds, the grid overlap must be updated so that the grid generator is called for each time step. For sufficiently large grid velocities, the low-density region behind the cylinder presents a problem for the Roe approximate Riemann solver used in the calculation of the numerical fluxes. This problem is known in the literature and other approximate Riemann solvers are under development to fix it.

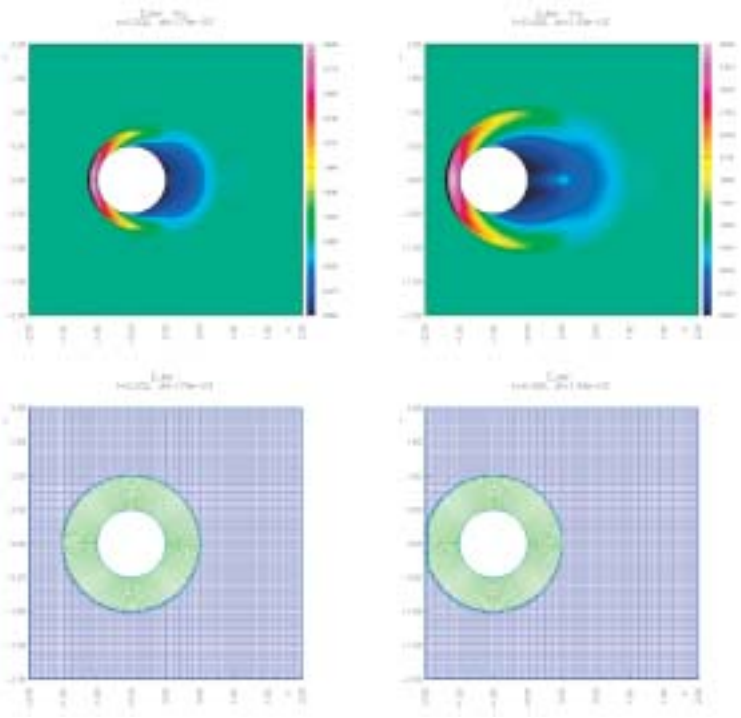

Fig. 3: Impulsive motion of a cylinder in a nonreactive flow.

Implicit Ocluders

Sinesio Pesco
Oregon Graduate Institutes
&
Claudio Silva
Oregon Graduate Institutes

ISCR Institute for Scientific Computing Research

Summary

Isosurfaces play a central role in the visualization of three-dimensional scalar fields. By being able to compute and display isosurfaces interactively, scientists can explore their datasets, study detailed features of interest, and obtain insights into the inner workings of real physical phenomena and simulated models. Because of their ubiquitous use in visualization, the computation and rendering of isosurfaces has received great attention from the visualization research community.

In this work, we propose a novel visibility-culling technique for optimizing the rendering and computation of opaque isosurfaces. Given a continuous scalar field $f(x)$ over a domain D and an isovalue w, our technique exploits the continuity of f to determine conservative visibility bounds implicitly, i.e., without the need for actually computing the isosurface. We generate implicit occluders based on the change in sign of $f^*(x) = f(x) - w$, from positive to negative (or vice versa) in the neighborhood of the isosurface. Consider, for example, the sign of f^* along a ray r cast from the current viewpoint. The first change in sign of f^* within D must contain an intersection of r with the isosurface. Any additional intersection of the isosurface with r is not visible. Implicit occluders constitute a general concept that can be exploited algorithmically in different ways depending on the framework adopted for visibility computations. In this work, we propose a simple "from-point" approach that exploits well-known hardware occlusion queries.

The implicit occluders project is in an advanced state of development. We first worked on a single resolution implementation that only worked for regular grids. Over the summer of 2003, we worked on extending this code to unstructured grids. While at LLNL, we have been working primarily on extending this work into an adaptive level-of-detail algorithm by extending the dual contouring work of Ju et al.

As of October 2003, I will be moving to the University of Utah. There, I will continue to work on this project. We would like to submit a paper on the implicit occluders work later this fall. Some options would be VisSym '04 or CGI (Computer Graphics International) '04. We are still deciding what material should be included in this paper, i.e., whether the adaptive work should be part of it, or simply send a revised version of the Visualization 2003 submission and leave the new adaptive work for a Visualization 2004 submission.